JOBSHEET 12

DOUBLE LINKED LIST

Mata Kuliah : Algoritma dan Struktur Data

Dosen : **Mungki Astiningrum, S.T., M.Kom.**

ILHAM DHARMA ATMAJA
TI 1A
(244107020220)

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNOLOGI INFORMASI
POLITEKNIK NEGERI MALANG TAHUN 2025

1. Praktikum 1 Double Linked List
    1) Class Node14

```
public class Node14 {
    int data;
    Node14 prev;
    Node14 next;

    public Node14(Node14 prev, int data, Node14 next) {
        this.prev = prev;
        this.data = data;
        this.next = next;
    }
}
```

2) Class DoubleLinkedList14

```
public class DoubleLinkedLists14 {
    Node14 head;
    int size;

    public DoubleLinkedLists14() {
        head = null;
        size = 0;
    }

    public boolean isEmpty() {
        return head == null;
    }

    public void addFirst(int item) {
        if (isEmpty()) {
            head = new Node14(null, item, null);
        } else {
            Node14 newNode = new Node14(null, item, head);
            head.prev = newNode;
            head = newNode;
        }
```

```
        }
        size++;
    }

    public void addLast(int item) {
        if (isEmpty()) {
            addFirst(item);
        } else {
            Node14 current = head;
            while (current.next != null) {
                current = current.next;
            }
            Node14 newNode = new Node14(current, item, null);
            current.next = newNode;
            size++;
        }
    }

    public void add(int item, int index) throws Exception {
        if (index < 0 || index > size) {
            throw new Exception("Nilai indeks di luar batas");
        }
        if (index == 0) {
            addFirst(item);
        } else {
            Node14 current = head;
            int i = 0;
            while (i < index) {
                current = current.next;
                i++;
            }
            if (current.prev == null) {
                Node14 newNode = new Node14(null, item, current);
                current.prev = newNode;
                head = newNode;
            } else {
                Node14 newNode = new Node14(current.prev, item, current);
                newNode.prev.next = newNode;
```

```
                newNode.prev.next = newNode;
                current.prev = newNode;
            }
            size++;
        }
    }

    public int size() {
        return size;
    }

    public void clear() {
        head = null;
        size = 0;
    }

    public void print() {
        if (!isEmpty()) {
            Node14 tmp = head;
            while (tmp != null) {
                System.out.print(tmp.data + "\t");
                tmp = tmp.next;
            }
            System.out.println("\nBerhasil diisi");
        } else {
            System.out.println("Linked list kosong");
        }
    }
}
```

3) Class DoublelinkedListMain14

```
public class DoubleLinkedListsMain14 {
    public static void main(String[] args) throws Exception {
        DoubleLinkedLists14 dll = new DoubleLinkedLists14();


        dll.print();
        System.out.println("size : " + dll.size());
        System.out.println("================================");
        dll.addFirst(3);
        dll.addFirst(4);
        dll.addFirst(7);
        dll.print();
        System.out.println("size : " + dll.size());
        System.out.println("================================");
        dll.add(40, 1);
        dll.print();
        System.out.println("size : " + dll.size());
        System.out.println("================================");
        System.out.println("Size : " + dll.size());
        dll.clear();
        dll.print();
    }
}
```

4) Verifikasi Hasil

```
Linked list kosong
size : 0
================================
7       4       3
Berhasil diisi
size : 3
================================
7       40      4       3
Berhasil diisi
size : 4
================================
Size : 4
Linked list kosong
PS C:\Users\ILHAM DHARMA A\OneDrive\Documents\SEME
```

1.2. PErtanyaan

1. **Single Linked List** hanya memiliki satu arah referensi (next), yaitu dari node sekarang ke node selanjutnya
   Double Linked List memiliki dua arah referensi (prev dan next), yaitu ke node sebelumnya dan node berikutnya.
2. Atribut next menyimpan referensi ke node berikutnya dalam list.
   Atribut prev menyimpan referensi ke node sebelumnya dalam list.
3. head = null: menunjukkan bahwa linked list masih kosong (tidak ada node yang ditunjuk sebagai kepala).
   size = 0: menunjukkan bahwa jumlah elemen dalam linked list adalah nol saat pertama kali dibuat.
4. Karena node baru tersebut akan berada di paling depan linked list, maka tidak ada node sebelumnya.
5. head.prev = newNode membuat hubungan dua arah antara node baru dan node lama
6. Karena node baru ini akan menjadi node terakhir (tidak ada node setelahnya), maka next = null.

2. Praktikum 2
   1) Tambahan pada Class DoubleLinkedList14

```
public void removeFirst() throws Exception {

        if (isEmpty()) {

            throw new Exception("Linked list masih kosong, tidak dapat
dihapus");

        } else if (size == 1) {

            head = null;

            size--;

            return;

        }

        head = head.next;

        head.prev = null;

        size--;

    }

    public void removeLast() throws Exception {

        if (isEmpty()) {

            throw new Exception("Linked list masih kosong, tidak dapat
dihapus");

        } else if (head.next == null) {

            removeFirst();

            return;

        }

        Node14 current = head;
```

```
        while (current.next != null) {
            current = current.next;
        }
        current.prev.next = null;
        size--;
    }
    public void remove(int index) throws Exception {
        if (isEmpty() || index >= size) {
            throw new Exception("Nilai indeks di luar batas");
        } else if (index == 0) {
            removeFirst();
            return;
        }

        Node14 current = head;
        int i = 0;
        while (i < index) {
            current = current.next;
            i++;
        }

        if (current.next == null) {
            current.prev.next = null;
        } else if (current.prev == null) {
            current = current.next;
            current.prev = null;
            head = current;
        } else {
            current.prev.next = current.next;
            current.next.prev = current.prev;
        }
        size--;
    }

}
```

2) Tambhan pada Class DoubleLinkedListMasin14

```
 dll.addLast(50);
        dll.addLast(40);
        dll.addLast(10);
        dll.addLast(20);
        dll.print();
        System.out.println("size : " + dll.size());
        System.out.println("==============================");

        dll.removeFirst();
        dll.print();
        System.out.println("size : " + dll.size());
        System.out.println("==============================");

        dll.removeLast();
        dll.print();
        System.out.println("size : " + dll.size());
        System.out.println("==============================");

        dll.remove(1);
        dll.print();
        System.out.println("size : " + dll.size());

    }
}
```

3) Verifikasi Hasil

```
50      40      10      20
Berhasil diisi
size : 4
==============================
40      10      20
Berhasil diisi
size : 3
==============================
40      10
Berhasil diisi
size : 2
==============================
40
Berhasil diisi
size : 1
==============================
==============================
```

2.2. Pertanyaan

1. head = head.next; : menggeser posisi head ke node berikutnya (menghapus node pertama).
   head.prev = null; : node yang baru menjadi head harus tidak memiliki node sebelumnya, jadi prev diset null.
2. Lakukan perulangan hingga current.next == null, karena ini menandakan node tersebut adalah node terakhir.
3. Tidak memperbarui pointer prev dan next dengan benar.
   Tidak mengecek kondisi khusus seperti node pertama/terakhir.
4. current.prev.next = current.next; menghubungkan node sebelum current ke node setelahnya.
   current.next.prev = current.prev; menghubungkan node sesudah current ke node sebelumnya.

3. Praktikum 3
   1) Tambahan Pada Class DoubleLinkedLists14

```
public int getFirst() throws Exception {
        if (isEmpty()) {
            throw new Exception("Linked list kosong");
        }
        return head.data;
    }
    public int getLast() throws Exception {
        if (isEmpty()) {
            throw new Exception("Linked list kosong");
        }
        Node14 current = head;
        while (current.next != null) {
            current = current.next;
        }
        return current.data;
    }
    public int get(int index) throws Exception {
        if (isEmpty() || index >= size) {
            throw new Exception("Linked list kosong atau indeks di luar batas");
        }
        Node14 current = head;
        for (int i = 0; i < index; i++) {
            current = current.next;
```

```
            current = current.next;
        }
        return current.data;
    }

}
```

2) Tambahan Pada Class Main

```
  dll.print();
        System.out.println("size : " + dll.size());
        System.out.println("==============================");
        dll.addFirst(3);
        dll.addLast(4);
        dll.addFirst(7);
        dll.print();
        System.out.println("size : " + dll.size());
        System.out.println("==============================");
        dll.add(40, 1);
        dll.print();
        System.out.println("size : " + dll.size());
        System.out.println("==============================");

        System.out.println("Data awal Linked List adalah : " + dll.getFirst());
        System.out.println("Data akhir Linked List adalah : " + dll.getLast());
        System.out.println("Data indeks ke-1 : " + dll.get(1));
    }
}
```

3) Verifikasi hasil

```
Berhasil diisi
size : 1
==============================
==============================
40
Berhasil diisi
size : 1
==============================
7        3        40       4
Berhasil diisi
size : 4
==============================
7        40       3        40       4
Berhasil diisi
size : 5
==============================
Data awal Linked List adalah : 7
Data akhir Linked List adalah : 4
Data indeks ke-1 : 40
```

3.2. Pertanyaan

1. Method size() mengembalikan jumlah node (elemen) yang saat ini ada dalam linked list.
2. Ubah logika dalam method add(), get(), remove() dengan mengurangi 1 dari nilai indeks pengguna.
3. **Double linked list:** memiliki pointer prev dan next, jadi lebih fleksibel untuk add di tengah karena bisa mundur dan maju
   Single linked list: hanya punya next, jadi sulit melakukan add di tengah tanpa navigasi dari awal.
4. Kode (a) mungkin hanya menggunakan next, cocok untuk single linked list.
   Kode (b) mungkin juga mengatur prev, sehingga hanya cocok untuk double linked list.

4. Tugas Praktikum

4.1. Tugas 1

1) Class QueueVaksin

```
public class QueueVaksin14 {

    Node14 head;

    int size;


    public QueueVaksin14() {

        head = null;

        size = 0;

    }


    public boolean isEmpty() {

        return head == null;

    }


    public void enqueue(String nama) {

        if (isEmpty()) {

            head = new Node14(null, nama, null);

        } else {

            Node14 current = head;

            while (current.next != null) {

                current = current.next;

            }

            Node14 newNode = new Node14(current, nama, null);

            current.next = newNode;

        }

        size++;
```

```
        System.out.println(nama + " masuk dalam antrian vaksin");
    }

    public void dequeue() {
        if (isEmpty()) {
            System.out.println("Antrian kosong");
        } else {
            System.out.println(head.nama + " telah selesai divaksin.");
            head = head.next;
            if (head != null) {
                head.prev = null;
            }
            size--;
        }
    }

    public void print() {
        if (isEmpty()) {
            System.out.println("Antrian kosong");
        } else {
            System.out.println("Daftar Antrian Vaksin:");
            Node14 current = head;
            while (current != null) {
                System.out.println("- " + current.nama);
                current = current.next;
            }
        }
        System.out.println("Jumlah antrian tersisa: " + size);
    }
}
```

2) Class Main

```
import java.util.Scanner;


public class MainVaksin14 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        QueueVaksin14 qv = new QueueVaksin14();
        int pilih;
        do {
            System.out.println("\n==============================");
            System.out.println("PENGOLAHAN DATA VAKSIN");
            System.out.println("==============================");
            System.out.println("1. Tambah Antrian");
            System.out.println("2. Hapus Antrian");
            System.out.println("3. Daftar Antrian");
            System.out.println("4. Keluar");
            System.out.print("Pilih menu: ");
            pilih = sc.nextInt();
            sc.nextLine(); // Buang newline

            switch (pilih) {
                case 1:
                    System.out.print("Masukkan nama: ");
                    String nama = sc.nextLine();
                    qv.enqueue(nama);
                    break;
                case 2:
                    qv.dequeue();
                    break;
                case 3:
                    qv.print();
                    break;
                case 4:
                    System.out.println("Terima kasih!");
                    break;
                default:
```

```
                default:
                    System.out.println("Pilihan tidak valid.");
            }
        } while (pilih != 4);
    }
}
```

4.2. Tugas 2

1) Class Film14

```
public class Film14 {
    int id;
    String judul;
    double rating;

    public Film14(int id, String judul, double rating) {
        this.id = id;
        this.judul = judul;
        this.rating = rating;
    }
}
```

2) Class NodeFIlm

```
public class NodeFilm14 {
    Film14 data;
    NodeFilm14 prev, next;

    public NodeFilm14(NodeFilm14 prev, Film14 data, NodeFilm14 next) {
        this.prev = prev;
        this.data = data;
        this.next = next;
    }
}
```

3) Class DoubleLinkedListFilm

```
public class DoubleLinkedListFilm14 {
    NodeFilm14 head;
    int size;

    public void addLast(Film14 data) {
        if (head == null) {
            head = new NodeFilm14(null, data, null);
        } else {
            NodeFilm14 current = head;
            while (current.next != null) {
                current = current.next;
            }
            NodeFilm14 newNode = new NodeFilm14(current, data, null);
            current.next = newNode;
        }
        size++;
    }

    public void print() {
        NodeFilm14 current = head;
        System.out.println("DAFTAR FILM:");
        while (current != null) {
            System.out.println("ID: " + current.data.id + " | Judul: " +
current.data.judul + " | Rating: " + current.data.rating);
            current = current.next;
        }
    }

    public void searchById(int id) {
        NodeFilm14 current = head;
        while (current != null) {
            if (current.data.id == id) {
                System.out.println("Ditemukan: ID: " + current.data.id + ",
Judul: " + current.data.judul + ", Rating: " + current.data.rating);
                return;
            }
```

```
            current = current.next;
        }
        System.out.println("Film dengan ID " + id + " tidak ditemukan.");
    }

    public void sortDescending() {
        if (head == null || head.next == null) return;

        for (NodeFilm14 i = head; i != null; i = i.next) {
            NodeFilm14 max = i;
            for (NodeFilm14 j = i.next; j != null; j = j.next) {
                if (j.data.rating > max.data.rating) {
                    max = j;
                }
            }

            if (max != i) {
                Film14 temp = i.data;
                i.data = max.data;
                max.data = temp;
            }
        }
    }
}
```

4) Main Film

```
import java.util.Scanner;


public class MainFilm14 {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      DoubleLinkedListFilm14 filmList = new DoubleLinkedListFilm14();
```

```
int pilihan;
 do {
    System.out.println("\n=== MENU FILM ===");
    System.out.println("1. Tambah Film");
    System.out.println("2. Cetak Daftar Film");
    System.out.println("3. Cari Film berdasarkan ID");
    System.out.println("4. Urutkan berdasarkan Rating (Descending)");
    System.out.println("5. Keluar");
    System.out.print("Pilih: ");
    pilihan = sc.nextInt();
    sc.nextLine(); // buang newline

    switch (pilihan) {
      case 1:
        System.out.print("ID: ");
        int id = sc.nextInt();
        sc.nextLine();
        System.out.print("Judul: ");
        String judul = sc.nextLine();
        System.out.print("Rating: ");
        double rating = sc.nextDouble();
        Film14 film = new Film14(id, judul, rating);
        filmList.addLast(film);
        break;
      case 2:
        filmList.print();
        break;
      case 3:
        System.out.print("Masukkan ID yang dicari: ");
        int cari = sc.nextInt();
        filmList.searchById(cari);
        break;
```

```
            case 4:
                filmList.sortDescending();
                filmList.print();
                break;
            case 5:
                System.out.println("Keluar...");
                break;
            default:
                System.out.println("Pilihan tidak valid.");
        }
    } while (pilihan != 5);
  }
}
```

4.3. Link Git Hub
https://github.com/ianmen10/SEMESTER-genap2.git